AutoCAD: Adaptación de un bloque ortonormal 2D [3D] a un paralelógramo [paralelepípedo] dado

NOGUÉS MAÑÉ, Carles

Universitat Politècnica de Catalunya, España
Departament d'Expressió Gràfica a l'Enginyeria
e-mail:nogues@ege.upc.edu

RESUMEN

Si se ha tenido la precaución de referir a un bloque 2D un cuadrado unitario ortogonal, será inmediato insertarlo de manera que éste se adapte a cualquier marco rectangular establecido en el dibujo. Sin embargo, la inmediatez desaparece si pretendemos encadenar inserciones de manera que, además de una combinación simple de escalado, giro y traslación, se halle implícita una transformación de cortadura: cuando el marco receptor sea un paralelógramo oblicuo. Porque está claro que si insertamos el bloque girado, convertimos la inserción en bloque y volvemos a insertar, esta vez con escalado no uniforme, la figura transformada del primitivo cuadrado de referencia será un paralelógramo, pero el problema es: dado un marco romboidal concreto, ¿con qué giro hay que realizar la primera inserción, y qué giro y factores de escala debemos aplicar a la segunda para que el cuadrado de referencia se ciña al marco? El problema se complica si queremos aprovechar el resultado de la primera inserción para adaptaciones a otros paralelógramos, creando un sistema no redundante de inserciones intermedias, pero se ha diseñado un programa AutoLISP que resuelve estas cuestiones en 2D, en 3D y puede operar con bloques con atributos, no sólo situados en el plano base sino ubicados y orientados libremente.

Palabras clave: AutoCAD: BLOQUE, INSERT. Programación: AutoLISP. Geometría: afinidad.

ABSTRACT

If we reference an orthogonal unit square to a 2D-block, the block can immediately be inserted in such a way that the orthogonal unit square adapts to any existing rectangular frame in the drawing. However, this immediacy is lost if we wish to perform a series of insertions in such a way that, in addition to a simple combination of scaling, rotating and translating, they implicitly include a shearing transformation when the receiving frame is an oblique parallelogram. This is because if we rotate the block and then insert it, convert the insertion into a block and reinsert (this time using a non-uniform scale factor), the figure resulting from this transformation of the original reference square will be a parallelogram. However, the question we wish to tackle is the following. Given a specific rhomboid frame, exactly what rotation do we need when making the first insertion, and what rotation and scale factors must we apply when inserting for the second time, in order to ensure that the reference square fits the frame exactly? The problem becomes more complicated if we wish to use the result of the first insertion and adapt it for other parallelograms, creating a non-redundant system of intermediate insertions. An AutoLISP program has been designed which solves these problems in 2D and in 3D. Furthermore, it can cope with blocks that do not sit squarely on the flat base, but which are arbitrarily located and orientated.

Key words: AutoCAD: BLOCK, INSERT. Software: AutoLISP. Geometry: affine transformations.

1. Introducción

Hace diez años me propuse abordar la cuestión planteada en el resumen precedente. La idea motriz del empeño fue la mejor adecuación del método gráfico llamado de Ritz (determinación de los ejes de una elipse a partir de dos diámetros conjugados) a la mecánica de inserción de los bloques AutoCAD, en relación a una resolución algebraica convencional basada en transformaciones lineales. Obtenidos unos resultados que desde la perspectiva actual calificaría de "manifiestamente mejorables", en fecha 28-01-97 me puse en contacto con Autodesk Spain para proponerles un trato: esa modesta aportación al desarrollo de aplicaciones para su producto estrella a cambio de licencias para regularizar la precaria cobertura legal bajo la que tenía que impartir docencia en la E.T.S. de Ingenieros de Telecomunicación de Barcelona. Como era de prever, no obtuve respuesta y la solución al problema de las licencias vino por otro lado; y como, con la consecución de unos mínimos objetivos, el tema ya no me estimulaba lo suficiente y dejó paso a otros, este trabajo quedó en hibernación. Años después lo he recuperado en el marco de un proyecto más ambicioso: una monografía que intentara cubrir el vacío bibliográfico que se detecta en la vertiente geométrica de la inserción de bloques, a diferencia de la vertiente estructura de datos (incrustación de dibujos via ***INSERT*** versus vinculación via ***REFX***), profusamente tratada. Con tal propósito se me ocurrió reunir en una misma publicación material docente sobre el tema (una parte de los apuntes de la asignatura *ELEMENTS DE CAD*, dedicados a sistematizar en tipologías geométricas distintos casos de inserción) y una versión actualizada (de la versión 13 a la 16), mejorada y comentada de aquel *software* que dormía el sueño de los justos en el limbo (metafórico, pues parece ser que ya no hay otro) de las innovaciones frustradas. La obra, en catalán, responde al siguiente plan:

INTRODUCCIÓ.
1- CAMPS D'INSERCIONS EN 2D: 5 EXERCICIS COMENTATS.
1.1- EXERCICI 1: escales X i Y en funció de coordenades cartesianes.
1.2- EXERCICI 2: escala única en funció de coordenades cartesianes.
1.3- EXERCICI 3: escala única i orientació en funció de coordenades polars (A).
1.4- EXERCICI 4: escala única i orientació en funció de coordenades polars (B).
1.5- EXERCICI 5: escales X, Y i orientació en funció de coordenades polars.
1.6- ... I LA TORNA.
2- ESMOLANT LES EINES AMB AUTOLISP.
2.1- Optimització dels recursos existents: 2 EXEMPLES.
2.2- Creació de nous recursos: ENCAIX 2D DE BLOCS SENSE ATRIBUTS.
2.3- ENCAIX 2D DE BLOCS AMB ATRIBUTS.
2.4- ENCAIX 3D DE BLOCS AMB ATRIBUTS.
2.5- ENCAIX 3D DE BLOCS AMB ATRIBUTS UBICATS I ORIENTATS LLIUREMENT.
2.6- ALTERNATIVES: redefinició de l'itinerari i repercussió sobre l'estructura de blocs auxiliars.
ANNEX: Plantejament algebraic de transformacions afins en 3D amb caràcter finalista.

Con un texto tan o más apretado que el empleado aquí, me temo que va a sobrepasar las 300 páginas: la 1ª parte y el capítulo 2.1 se llevaron 70, y por lo realizado (la exposición progresa en paralelo por los capítulos 2.4 y 2.5, aunque sólo el *software* del capítulo 2.6 está por resolver) calculo que el desarrollo del tema propuesto (capítulos 2.2 al 2.6, más el anexo) representará al menos otras 250.

2. Contenido

Dado el carácter de comunicación de los materiales a presentar en el Congreso i la lógica acotación de su tamaño, el dilema entre glosar un trabajo o presentar el trabajo mismo está servido: la última opción sólo es viable para los de reducida extensión o que admitan expresar su quintaesencia en gráficos, fórmulas o algoritmos. Tratar de resumir con un mínimo de rigor una línea argumental desarrollada a lo largo de las anunciadas más de 250 págs. parecía una empresa difícil y sobre todo estéril. He creído más acertado emplazar a finales de año (fecha estimada para la conclusión del trabajo) a quienes estén interesados, y ofrecerles ahora la primicia: el programa AutoLISP en que ha culminado el discurso (por lo menos hasta el capítulo 2.5, a falta de las mejoras previstas en 2.6). Con ello persigo el doble propósito de poner a disposición de todo el mundo una herramienta útil y de que la herramienta quede almacenada en el CD que editará el Congreso, para así poder desmentir eventuales usurpaciones de

autoría cuando le dé una difusión más amplia. Si no hubiera sido por la decisión de última hora de los organizadores de recortar la extensión máxima de las comunicaciones desde 20 hasta 15 páginas, mi intención era presentar el **alfa** y la **omega** para mostrar hasta qué punto la complejidad del programa se había disparado por cuestiones relativamente accesorias como la de los atributos (y no digamos, con la extrapolación a atributos colocados arbitrariamente): la versión inicial de página y media (capítulo 2.2) y las casi 14 páginas de la más reciente (capítulo 2.5); pude hacerlo cumpliendo escrupulosamente la letra de la norma, sólo con escribir el código de corrido (sustituir con espacios simples los cambios de línea con sangrado que tanto ayudan a apreciar su estructura lógica), pero habría sido una salvajada y he preferido limitarme a la **omega**:

```
;; (vl-load-com)
;; (vlr-remove-all)
;; (vlr-COMMAND-reactor ()'((:vlr-CommandWillStart . -ECO-1)
;;                          (:vlr-CommandEnded . -ECO-2)))

;; (defun -ECO-1 (N-REACTIU L-ORDRE)
;;   (if CTRL--ECO (BS (1+ (strlen (getcname (strcat "_" (car L-ORDRE))))))))

;; (defun -ECO-2 (N-REACTIU L-ORDRE)
;;   (if CTRL--ECO (BS (1+ (strlen CTRL--ECO)))))

;; (defun SENSE-RASTRE ()
;;   (command "DESHACER" (progn (if (= ECO 1)
;;                                  (progn
;;                                    (BS 100)
;;                                    (setq CTRL--ECO "I")))
;;                              "I"))
;;   (if (= ECO 1)
;;       (progn
;;         (BLANC)
;;         (setq CTRL--ECO ()))))

(defun QUASI-SENSE-RASTRE ()
  (command "DESHACER" (progn (if (= ECO 1) (BS 100)) "I"))
  (if (= ECO 1) (BLANC)))

(defun C:INS2D () (INSERTOK T))

(defun C:INS3D () (INSERTOK ()))

(defun BS (I) (repeat I (princ "\10 \10")))

(defun BLANC ()
  (princ "\r") (repeat 100 (princ " ")) (princ "\r") (princ))

(defun REFX ()
  (strcat "\"" BLOC "\"\nNo se puede insertar con INSERT ni INSERTOK\nporque "
          (if (= (logand (cdr (assoc 70 O)) 16) 0) "es" "depende de")
          " una referencia externa."))

(defun RUTES (/ MS PREFIX N C)
  (setq MS (strcat "\"" BLOC
             ".dwg\"\nNo se encuentra el archivo en el camino de búsqueda:\n  "
             (substr (findfile "ACAD.EXE") 1 (- (strlen (findfile "ACAD.EXE")) 8))
             " (directorio actual)\n  " (getvar "DWGPREFIX") "\n  ")
        PREFIX (getvar "ACADPREFIX") N 0)
```

```
  (repeat (strlen PREFIX)
          (setq N (1+ N) C (substr PREFIX N 1)
                MS (strcat MS (if (= C ";") "\n  " C))))
  (substr MS 1 (- (strlen MS) 3)))

(defun DIBUIX (/ A B)
  (command "INSERT" BLOC ())
  (setq A (tblnext "BLOCK" T))
  (while (setq B (tblnext "BLOCK")) (setq A B))
  (strcase (cdr (assoc 2 A))))

(defun DEFECTE (TXT) (if (= TXT "") "" (strcat " <" TXT ">")))

(defun U*V (U V)
  (list (- (* (cadr U) (last V)) (* (last U) (cadr V)))
        (- (* (last U) (car  V)) (* (car  U) (last V)))
        (- (* (car  U) (cadr V)) (* (cadr U) (car  V)))))

(defun VATR (W* N* M* VD*)
  (setq W W* N N* M M* VD VD*
        V (if ATREQ
              (getstring (strcat "\n" (if (= M "") N M) (DEFECTE VD) ": ") T)
              "")
        V (if (= V "") VD V)))

(defun VAL-ATRIBS-1 (LA / N M V)
  (if (= (logand ICVP 8) 0)
      (progn
        (if (and ATREQ (not INI))
            (setq INI (not (prompt "\nIndique valores de atributo"))))
        (VATR (= (logand ICVP 4) 4) (cdr (assoc 2 LA))
              (cdr (assoc 3 LA)) (cdr (assoc 1 LA)))
        (setq LLAA (cons (list W N M V) LLAA)))))

(defun VAL-ATRIBS-2 (/ W1 N M V)
  (setq INI () W ())
  (foreach LA (reverse LLAA)
    (if (not EXT)
        (setq W1 (car LA) LA (cdr LA)))
    (if (car LA)
        (progn
          (if (and ATREQ (not INI))
              (setq INI (not (prompt "\nVerificar valores de atributo"))))
          (VATR (if (or W1 (and EXT (not PRED))) (cons "" W) W)
                (cadr LA) (caddr LA) (last LA)))
        (setq V (last LA)))
    (setq V (list V))
    (if EXT
        (setq WWAA (append WWAA V))
        (if W1
            (setq WWAA-1 (append WWAA-1 V))
            (setq WWAA-2 (append WWAA-2 V)))))
  (if EXT
      (if PRED (repeat WW (setq W (cons "" W))))
      (setq E1 (strcat (itoa (length W)) E1)))
  (setq WW W))
```

```
(defun VAL-ATRIBS (OO EXT PRED)
  (foreach O (reverse OO)
    (setq ICVP (cdr (assoc 70 O)))
    (if (and PRED (or (not O) (= (logand ICVP 8) 8)))
        (setq LLAA (cons () LLAA))
        (VAL-ATRIBS-1 O)))
  (VAL-ATRIBS-2))

(defun MAKEBLOC (BLOC/2 ATRIBS OO)
  (entmake (list '(0 . "BLOCK") (cons 2 BLOC/2) '(10 0 0 0)
                 (cons 70 (if ATRIBS 2 0))))
  (foreach O (reverse OO) (entmake O))
  (entmake '((0 . "ENDBLK"))))

(defun MNLOC (O)
  (MAKEBLOC (setq BLOC NLOC-1) (or AT-C (if O OO-2)) (append OO-2 OO-1))
  (MAKEBLOC NLOC-2 () OO-3))

(defun MBLOC (O)
  (MAKEBLOC BLOC-1 AT-C OO-1)
  (MAKEBLOC BLOC-2 (if O OO-2) (setq OO-2 (append OO-3 OO-2))))

(defun LINIA-BASE ()
  (polar (cdr (assoc 11 LE))
         (- (cdr (assoc 50 LE)) PI/2)
         (* (cdr (assoc 40 LE))
            (if (= C-74 1)
                (/ 1.0 -3)
                (if (= C-74 2) 0.5 1)))))

(defun PUNT (P) (list '(0 . "POINT") (cons 10 (trans P Z 0)) (cons 210 Z)))

(defun INSPARCIAL (EX EY EZ ANG)
  (command "INSERT" BLOC-1 O "XYZ" EX EY EZ ANG)
  (setq SS (ssadd (entlast)))
  (if OO-2
      (progn
        (command "ATTREQ" 0
                 "INSERT" BLOC-2 O "XYZ" EX EY EZ ANG
                 "ATTREQ" (if ATREQ 1 0)
                 "DESCOMP" (entlast))
        (setq SA (ssget "P") K -1)
        (while (setq E (ssname SA (setq K (1+ K))))
               (ssadd E SS)))))

(defun XOR (A B) (and (not (and A B)) (or A B)))

(defun B->N (A B / C)
  (setq C (= B (if A BLOC-1 NLOC-1))
        E (cdr (assoc -2 (tblsearch "BLOCK" B))))
  (if (/= (cdr (assoc 0 (entget E))) "ENDBLK")
      (while E
             (setq LE (entget E) AT (= (cdr (assoc 0 LE)) "ATTDEF"))
             (if (and AT (or (and A (not C))
                             (and (not A) C (= (logand (cdr (assoc 70 LE)) 2) 0))))
```

```
                (setq AT-C T OO-2 (cons LE OO-2))
                (if C
                    (setq AT-C (if AT-C T AT)
                          OO-1 (cons LE OO-1))
                    (setq OO-3 (cons LE OO-3))))
            (setq E (entnext E)))))

(defun ATRIB-ATIPIC (I)
  (tblsearch "BLOCK" (strcat "ATRIBATIP_" (itoa (set I (1+ (eval I)))) "_DE_" BLOC)))

(defun BLOC->WWAA (/ EE OO EEOO EO OORD N)
  (setq OO (reverse OO-1)
        E1 (car OO) E2 (cadr OO)
        E1 (read (strcat "(" (cdr (assoc 1 E1)) ")"))
        WW (car E1) E1 (cdr E1)
        E2 (read (strcat "(" (cdr (assoc 1 E2)) ")")) N -1
        EE (repeat K (setq EE (cons (nth (setq N (1+ N)) E2) EE)))
        E2 (reverse EE)
        EE (append E1 E2) OO () N 0
        OO (repeat K (setq OO (cons (entget (cdr (assoc -2 (ATRIB-ATIPIC 'N)))) OO)))
        OO (reverse (append OO OO-2))
        EEOO (mapcar 'cons EE OO) N -1)
  (repeat (+ (length OO-2) J)
          (setq N (1+ N)
                EO (cdr (assoc N EEOO))
                OORD (cons EO OORD)))
  (VAL-ATRIBS OORD T T)
  (foreach E E1
           (if (setq N (nth E WWAA))
               (setq WWAA-1 (append WWAA-1 (list N)))))
  (foreach E E2
           (if (setq N (nth E WWAA))
               (setq WWAA-2 (append WWAA-2 (list N))))))

(defun WWAA->BLOC (/ O1 O2)
  (setq O1 (last OO-1))
  (if (and (= (cdr (assoc 0 O1)) "ATTDEF")
           (= (cdr (assoc 2 O1)) "WWAA-1")
           (= (cdr (assoc 70 O1)) 3))
      (setq OO-1 (reverse (cdr (cdr (reverse OO-1))))))
  (if OO-4
      (setq O1 (list '(0 . "ATTDEF") '(8 . "0") '(70 . 3) '(2 . "WWAA-1")
                     '(3 . "") (cons 1 E1) '(72 . 0) '(74 . 0) '(10 0 0 0)
                     '(210 0 0 1) '(7 . "Standard") '(40 . 1) '(41 . 1))
            O2 (subst (cons 1 E2) (cons 1 E1)
                      (subst '(2 . "WWAA-2") '(2 . "WWAA-1")
                             O1))
            OO-1 (append OO-1 (list O2 O1)))))

(defun SEGR-PUNTS (/ E)
  (setq SA (ssadd))
  (repeat (* 3 OO-4)
          (setq E (ssname SS 1))
          (ssadd E SA)
          (ssdel E SS)))
```

```
(defun PRO-DESHACER-I/F ()
  (if OO-2
      (eval (append '(command "BLOQUE" BLOC-2*)
                    (if (tblsearch "BLOCK" BLOC-2*) '("S"))
                    '(O (ssdel E SS) "")))
      (progn
        (entmake (list '(0 . "BLOCK") (cons 2 BLOC-2*) (cons 10 O) '(70 . 0)))
        (entmake '((0 . "ENDBLK"))))))

(defun REDEF-BLOC*S (/ LB)
  (setvar "CLAYER" "0")
  (setvar "CECOLOR" "PORBLOQUE")
  (setvar "CELTYPE" "PORBLOQUE")
  (setvar "CELWEIGHT" -2)
  (if BL1EX
      (progn
        (tblnext "BLOCK" T)
        (while (setq LB (tblnext "BLOCK"))
               (if (wcmatch (setq BLOC-1* (cdr (assoc 2 LB)))
                            (strcat BLOC-1 M (substr M 2)))
                   (progn
                     (setq E (cdr (assoc -2 LB))
                           LE (entget E)
                           K (cdr (assoc 41 LE))
                           BLOC-2* (strcat BLOC-2
                                           (substr BLOC-1* (1+ (strlen BLOC-1)))))
                     (command "SCP" "EZ" O (mapcar '+ O (cdr (assoc 210 LE))))
                     (INSPARCIAL K K K (/ (* (cdr (assoc 50 LE)) 180) PI))
                     (eval (append '(command "SCP" "PR"
                                             "BLOQUE" BLOC-1* "S" O
                                                      (setq E (ssname SS 0)) "")))
                     (PRO-DESHACER-I/F))))
        (setq BLOC-1* BLOC-1 BLOC-2* BLOC-2)
        (tblnext "BLOCK" T)
        (while (setq LB (tblnext "BLOCK"))
               (if (wcmatch (setq BLOC* (cdr (assoc 2 LB)))
                            (strcat N "*"))
                   (progn
                     (setq E (cdr (assoc -2 LB))
                           LE (entget E)
                           J (if (wcmatch BLOC* N)
                                 ""
                                 (substr BLOC* (1+ (strlen BL))
                                               (1- (* 2 (strlen M)))))
                           BLOC-1 (strcat BLOC-1* J)
                           BLOC-2 (strcat BLOC-2* J))
                     (if (> J "") (command "SCP" "EZ" O
                                           (mapcar '+ O (cdr (assoc 210 LE)))))
                     (INSPARCIAL (cdr (assoc 41 LE))
                                 (cdr (assoc 42 LE))
                                 (cdr (assoc 43 LE))
                                 (/ (* (cdr (assoc 50 LE)) 180) PI))
                     (if (> J "") (command "SCP" "PR"))
                     (if OO-4 (SEGR-PUNTS))
                     (command "BLOQUE" BLOC* "S" O SS "")
```

```
                    (if OO-4
                        (progn
                          (setq K (strcat "ATRIBSATIPS_DE_" BLOC*))
                          (eval (append '(command "BLOQUE" K)
                                                (if (tblsearch "BLOCK" K)
                                                    '("S"))
                                                '(O SA ""))))))))
        (setq BLOC-1 BLOC-1* BLOC-2 BLOC-2*)))
  (if OO-4
      (progn
        (setq BLOC-1* BLOC-1 BLOC-2* BLOC-2 OO-2 T J 0)
        (setvar "ATTREQ" 0)
        (repeat OO-4
                (setq J (1+ J) BLOC-2 (strcat "ATRIBATIP_" (itoa J) "_DE_" BL))
                (tblnext "BLOCK" T)
                (while (setq LB (tblnext "BLOCK"))
                       (if (wcmatch (setq BLOC* (cdr (assoc 2 LB)))
                                    (strcat BLOC-2 M))
                           (progn
                             (setq BLOC-1 "BLOC_NUL"
                                   E (cdr (assoc -2 LB)) LE (entget E)
                                   K (cdr (assoc 41 LE)))
                             (INSPARCIAL K K K (/ (* (cdr (assoc 50 LE)) 180) PI))
                             (command "BLOQUE" BLOC* "S" 0 SS ""))))))
        (setq BLOC-1 BLOC-1* BLOC-2 BLOC-2*)))
  (setvar "ATTREQ" (if ATREQ 1 0))
  (setvar "CLAYER" CAPA))

(defun J-PUNTS (LB)
  (setq E (cdr (assoc -2 LB)) J 0)
  (while E
         (setq J (if (= (cdr (assoc 0 (entget E))) "POINT") (1+ J) J)
               E (entnext E)))
  J)

(defun AVIS ()
  (alert (strcat "ATENCIÓN:\n\nComo \"" BLOC "\" se creó con la orden BLOQUE y no "
                 "con BLOQUEOK,\n\nlos atributos no justificados sobre la línea "
                 "base pueden aparecer movidos.")))

(defun AJUSTA-E (E)
  (if (= E "")
      ""
      (progn
        (setq OZ (mapcar '1+ (read (strcat "(" E ")")))
              E "")
        (foreach Z OZ (setq E (strcat E " " (itoa Z)))))))

(defun SEGR-ATRIBS (/ NORM NLOC-1 NLOC-2 NL1EX NL2EX BL1EX BL2EX BLOK AT AT-C NE
                      E1 E2 EXT INI LLAA C-10 C-11 C-72 C-74 OO-3 OZ Z)
  (setq NORM (and NORM-XY NORM-Z)
        NLOC-1 (strcat BLOC "_AMB_ATRIBSTIPS")
        NL1EX (tblsearch "BLOCK" NLOC-1)
        NLOC-2 (strcat "ATRIBSATIPS_DE_" BLOC)
        NL2EX (tblsearch "BLOCK" NLOC-2)
        BLOC-1 (strcat BLOC "_SENSE_ATRIBS")
```

```
      BL1EX (tblsearch "BLOCK" BLOC-1)
      BLOC-2 (strcat "ATRIBS_DE_" BLOC)
      BL2EX (tblsearch "BLOCK" BLOC-2) K 0
      OO-4 (while (ATRIB-ATIPIC 'K) K)
      K (if NL2EX
            (J-PUNTS NL2EX)
            (if BL2EX
                (J-PUNTS BL2EX)))
      K (if (> K 0) (/ K 3)) J OO-4)
(if (< K OO-4)
    (setq OO-4 K)
    (if (> K OO-4)
        (progn
          (setq NL2EX () BL2EX ())
          (command "LIMPIA" "B" (strcat NLOC-2 "," BLOC-2) "N"))))
(if (and (or NL2EX BL2EX) (not (XOR NL1EX NL2EX)) (not (XOR BL1EX BL2EX)))
    (progn
      (if (and (setq LE (entget (cdr (assoc -2 (if BL1EX BL2EX NL2EX))))
                     E (/= (cdr (assoc 0 LE)) "ENDBLK"))
               (= (cdr (assoc 0 LE)) "ATTDEF")
               (= (cdr (assoc 2 LE)) "NO-BLOQUEOK"))
          (AVIS))
      (if BL1EX
          (progn
            (B->N T BLOC-1)
            (B->N T BLOC-2))
          (progn
            (B->N () NLOC-1)
            (B->N () NLOC-2)))
      (if OO-3
          (BLOC->WWAA)
          (if OO-2
              (progn
                (VAL-ATRIBS OO-2 T ())
                (setq WWAA-1 WWAA))))
      (if (and (not NL1EX) NORM)
          (MNLOC ())
          (if (not (or BL1EX NORM))
              (MBLOC ())
              (if NORM
                  (setq BLOC NLOC-1)))))
    (progn
      (setq  E (cdr (assoc -2 (tblsearch "BLOCK" BLOC)))
             BLOK E OO-4 () NE -1 E1 "" E2 "")
      (while E
             (setq LE (entget E))
             (if (equal E BLOK)
                 (setq BLOK (if (and (= (cdr (assoc 0 LE)) "ATTDEF")
                                     (= (cdr (assoc 2 LE)) "BLOQUEOK")) 2 0)))
             (if (> BLOK 1)
                 (setq BLOK 1)
                 (if (and (setq AT (= (cdr (assoc 0 LE)) "ATTDEF"))
                          (= (logand (setq ICVP (cdr (assoc 70 LE))) 2) 0))
                     (progn
                       (setq NE (1+ NE)
                             C-72 (cdr (assoc 72 LE))
```

```
      C-74 (cdr (assoc 74 LE))
      BLOK (if (and (= BLOK 0)
                    (or (= C-72 4)
                    (and (< C-72 3) (> C-74 0))))
               (progn
                 (setq E1 (AJUSTA-E E1)
                       E2 (AJUSTA-E E2)
                       E1 (strcat E1 " " (itoa NE))
                       NE (1+ NE)
                       OO-2
                       (append OO-2
                         (list
                           (list '(0 . "ATTDEF")
                                 '(8 . "0") '(70 . 9)
                                 '(2 . "NO-BLOQUEOK")
                                 '(3 . "") '(1 . "")
                                 '(72 . 0) '(74 . 0)
                                 '(10 0 0 0)
                                 '(210 0 0 1)
                                 (assoc 7 LE)
                                 (assoc 40 LE)
                                 (assoc 41 LE)))))
                 (AVIS))
               BLOK)
      LE (if (= C-72 4)
             (subst (cons 72 (setq C-72 1))
                    (cons 72 4)
                    (subst (cons 74 (setq C-74 2))
                           (assoc 74 LE)
                           LE))
             LE)
      LE (if (and (< C-72 3) (> C-74 0))
             (subst (cons 74 0)
                    (assoc 74 LE)
                    (subst (cons 11 (LINIA-BASE))
                           (assoc 11 LE)
                           LE))
             LE)
      OZ (last (cdr (assoc 10 LE)))
      Z (cdr (assoc 210 LE)))
(VAL-ATRIBS-1 LE)
(if (and (equal OZ 0 Q0) (equal Z '(0 0 1) Q0))
    (setq LLAA (if (= (logand ICVP 8) 0)
                   (cons (cons T (car LLAA)) (cdr LLAA))
                   LLAA)
          E1 (strcat E1 " " (itoa NE))
          OO-2 (cons LE OO-2))
    (setq LLAA (if (= (logand ICVP 8) 0)
                   (cons (cons () (car LLAA)) (cdr LLAA))
                   LLAA)
          E2 (strcat E2 " " (itoa NE))
          OO-3 (cons (PUNT (list 0 0 OZ)) OO-3)
          OO-3 (cons (PUNT (list 1 0 OZ)) OO-3)
          OO-3 (cons (PUNT (list 0 1 OZ)) OO-3)
          C-10 (assoc 10 LE)
          C-11 (assoc 11 LE)
```

```
                                OO-4 (cons (subst (list 10 (cadr C-10)
                                                           (caddr C-10) 0)
                                                   C-10
                                                   (subst (list 11 (cadr C-11)
                                                                  (caddr C-11) 0)
                                                                  C-11
                                                                  (subst '(210 0 0 1)
                                                                         (assoc 210 LE)
                                                                         LE)))
                                           OO-4))))
                      (setq OO-1 (cons LE OO-1)
                            AT-C (if AT-C T AT))))
              (setq E (entnext E)))
        (VAL-ATRIBS-2)
        (if (or OO-4 (not NORM))
            (progn
              (command "REGENT"
                       "CECOLOR" "PORCAPA"
                       "CELTYPE" "PORCAPA"
                       "CELWEIGHT" -1)
              (WWAA->BLOC)
              (if (or NL1EX NL2EX NORM) (MNLOC T))
              (if (or BL1EX BL2EX (not NORM)) (MBLOC T))
              (if OO-4
                  (progn
                    (if (not (tblsearch "BLOCK" "BLOC_NUL"))
                        (MAKEBLOC "BLOC_NUL" ()()))
                    (setq J 0)
                    (foreach O (reverse OO-4)
                      (MAKEBLOC (strcat "ATRIBATIP_" (itoa (setq J (1+ J)))
                                        "_DE_" BL) T (list O)))
                    (setq OO-4 (length OO-4))))
              (if (or BL1EX OO-4) (REDEF-BLOC*S))
              (if (not (atom OO-4)) (setq OO-4 (length OO-4)))
              (setvar "CECOLOR" COL)
              (setvar "CELTYPE" TLIN)
              (setvar "CELWEIGHT" GLIN))))))

(defun CALCULA (U V OV AGUT U* U** OU* OU** V**)
  (setq A (polar O (- (angle O V) PI/2) OV)
        W (if AGUT (angle A U) (angle U A))
        B (mapcar '/ (mapcar '+ A U) '(2 2 2))
        W (angle O (polar B W (distance O B))))
  (set U* (inters U (polar U W 1) O B ()))
  (set U** (inters O (polar O (+ W PI/2) 1) U (eval U*) ()))
  (set OU* (distance O (eval U*)))
  (set OU** (distance O (eval U**)))
  (set V** (inters V (polar V W 1) O (eval U**) ())))

(defun FIX+ (M / N) (itoa (if (> (- M (setq N (fix M))) 0.5) (1+ N) N)))

(defun INSERT** (BLOC X Y Z G / JJ KK SSAA)
  (command "INSERT" (strcat "ATRIBSATIPS_DE_" BLOC) O "XYZ" X Y Z G
           "DESCOMP" (entlast)
           "SCP" "")
  (setq SSAA (ssget "P") JJ 0 KK -1)
```

```
  (repeat OO-4
          (setq JJ (1+ JJ) BLOC (strcat "ATRIBATIP_" (itoa JJ) "_DE_" BL)
                LE (entget (cdr (assoc -2 (tblsearch "BLOCK" BLOC))))
                ICVP (cdr (assoc 70 LE))
                WWAA-1 (if (= (logand ICVP 8) 0) (list (car WWAA-2)))
                WWAA-2 (if WWAA-1 (cdr WWAA-2) WWAA-2)
                WW (if (and WWAA-1 (= (logand ICVP 4) 4)) '(""))
                KK (1+ KK) O (cdr (assoc 10 (entget (ssname SSAA KK))))
                KK (1+ KK) X (cdr (assoc 10 (entget (ssname SSAA KK))))
                KK (1+ KK) Y (cdr (assoc 10 (entget (ssname SSAA KK)))))
          (command "SCP" "3" O X Y)
          (setq OX (distance O X) OY (distance O Y) OZ OX
                O '(0 0 0) X (list OX 0 0) Y (trans Y 0 1) Z (U*V X Y)
                Z-XY O I (if (< (last Z) 0) -1 1)
                BLOC-1 "BLOC_NUL" BLOC-2 BLOC
                NORM-XY (equal (setq X-Y (expt (distance X Y) 2))
                               (setq X-O-Y (+ (expt OX 2) (expt OY 2))) Q0)
                NORM-Z T OO-2 T OO-4 ())
          (if NORM-XY
              (eval (append '(command "INSERT" BLOC O OX OY 0)
                            (if ATREQ (append WWAA-1 WW))))
              (INSERT*))
          (command "SCP" "PR"))
  (command "SCP" "PR"
           "BORRA" SSAA ""))

(defun INSERT* (/ X* X** Y* Y** Z** OX* OX** ZZ** Z-XY* XY XY* XY** OXY* OXY** BLAA)
  (command "CLAYER" "0"
           "CECOLOR" "PORBLOQUE"
           "CELTYPE" "PORBLOQUE"
           "CELWEIGHT" -2)
  (if NORM-XY
      (setq X* X X** X
            Y* (polar O PI/2 OX)
            Y** O OX* OX OX** OX
            Z-XY* (list (car Z) (* (cadr Z) (/ OX OY)) 0))
      (progn
        (CALCULA X Y OY (< X-Y X-O-Y) 'X* 'X** 'OX* 'OX** 'Y**)
        (setq Y* (polar O (+ (angle O X*) PI/2) OX*)
              A (inters Z-XY (polar Z-XY W 1) O X** ())
              Z-XY* (polar A (angle A Z-XY)
                           (/ (* (distance A Z-XY) (distance X* X**))
                              (distance X X**))))))
  (if NORM-Z
      (setq BLOC* (strcat BLOC "_"))
      (progn
        (setq X (inters X* Y* O Z-XY* ())
              X (if X X (if (< (car Z-XY*) 0)
                            (list (- (car Z-XY*)) (- (cadr Z-XY*)) 0)
                            Z-XY*))
              Z-XY (- (angle O X) (angle Y* X*))
              Z-XY (if (< Z-XY 0) (+ 2*PI Z-XY) Z-XY)
              Z-XY (if (or (equal Z-XY 0 Q0) (equal Z-XY 2*PI Q0)) 0 Z-XY)
              Z (trans (list (car Z-XY*) (cadr Z-XY*) (last Z)) 1 0))
        (command "SCP" "Z" O X
                 "SCP" "X" (* I 90))
```

```
      (setq XY (list OX* 0 0)
            Z (trans Z 0 1))
      (CALCULA XY Z (distance O Z) (> (car Z) 0) 'XY* 'XY** 'OXY* 'OXY** 'Z**)
      (setq ZZ** (distance Z Z**)
            K (strcat "_" (FIX+ (/ Z-XY PI Q0)) (FIX+ (/ OXY** OXY* Q0)))
            BLOC* (strcat BLOC K)
            BLOC-1* (strcat BLOC-1 K) BLOC-2* (strcat BLOC-2 K)
            BL*EX (and (tblsearch "BLOCK" BLOC-1*) (tblsearch "BLOCK" BLOC-2*)))
      (command "SCP" "PR" "SCP" "PR")
      (if (not BL*EX)
          (progn
            (INSPARCIAL (setq K (/ OXY* OXY**)) K K X*)
            (eval (append '(command "SCP" "Z" O X
                                    "SCP" "X" 90
                                    "GIRA" SS "" O XY*
                                    "SCP" "Z" O XY**
                                    "BLOQUE" BLOC-1*)
                                             (if (tblsearch "BLOCK" BLOC-1*)
                                                 '("S"))
                                             '(O (setq E (ssname SS 0)) "")))
            (PRO-DESHACER-I/F)
            (command "SCP" "PR")))
      (setq B BL*EX BLOC-1 BLOC-1* BLOC-2 BLOC-2*
            BLOC* (strcat BLOC* "_" (FIX+ (/ ZZ** OXY** Q0))))))
(if (not NORM-XY) (setq BLOC* (strcat BLOC* (FIX+ (/ OX** OX* Q0)))))
(if OO-4 (setq BLAA (strcat "ATRIBSATIPS_DE_" BLOC*)))
(if (or NORM-Z BL*EX) (setq BL*EX (and (tblsearch "BLOCK" BLOC*)
                                       (if OO-4 (tblsearch "BLOCK" BLAA) T))))
(if (not BL*EX)
    (progn
      (if NORM-Z
          (INSPARCIAL (setq K (/ OX* OX**)) K I X*)
          (progn
            (if B (command "SCP" "Z" O X
                           "SCP" "X" 90))
            (INSPARCIAL (setq K (/ 1 OX**)
                              E (* OXY** K)) (* ZZ** K) (* E (/ OX* OXY*)) XY**)
            (command "SCP" "PR" "SCP" "PR")))
      (if OO-4 (SEGR-PUNTS))
      (eval (append '(command "SCP" "Z" O X**
                              "BLOQUE" BLOC*)
                                       (if (tblsearch "BLOCK" BLOC*) '("S"))
                                      '(O SS "")))
      (if OO-4
          (eval (append '(command "BLOQUE" BLAA)
                                           (if (tblsearch "BLOCK" BLAA) '("S"))
                                          '(O SA ""))))
      (command "SCP" "PR")))
(command "CELWEIGHT" GLIN
         "CELTYPE" TLIN
         "CECOLOR" COL
         "CLAYER" CAPA)
(eval (append '(command "INSERT" BLOC* O "XYZ" OX** (distance Y Y**)
                        (* (if NORM-Z OZ OX**) I) X**)
              (if ATREQ (append WWAA-1 WW))))
(if OO-4 (INSERT** BLOC* OX** (distance Y Y**) (* (if NORM-Z OZ OX**) I) X**)))
```

```
(defun INSERTOK (2D / Q0 2*PI PI/2 M N CAPA COL TLIN GLIN ECO CTRL--ECO OSN ATDIA
                      ATREQ BL BLOC BLOC* BLOC-1 BLOC-2 BLOC-1* BLOC-2* BL*EX ICVP
                      WWAA WWAA-1 WWAA-2 WW O X Y Z UV Z-XY OX OY OZ X-Y X-O-Y A B
                      E LE I J K W NORM-XY NORM-Z SA SS OO-1 OO-2 OO-4)
  (setq Q0 0.001 M "_")
  (repeat (strlen (itoa (fix (/ 0.1 Q0)))) (setq M (strcat M "#")))
  (setq 2*PI (* PI 2) PI/2 (/ PI 2)
        CAPA (getvar "CLAYER")
        COL (getvar "CECOLOR")
        TLIN (getvar "CELTYPE")
        GLIN (getvar "CELWEIGHT")
        ECO (getvar "CMDECHO")
        OSN (getvar "OSMODE")
        ATDIA (getvar "ATTDIA")
        ATREQ (= (getvar "ATTREQ") 1)
        O (getvar "INSNAME")
        BLOC (getstring (strcat "\nIndique nombre de bloque" (DEFECTE O) ": ") T)
        BLOC (if (= BLOC "")
                 (if (> O "") O (prompt "\nNombre de bloque no válido."))
                 (if (or (and (setq O (tblsearch "BLOCK" BLOC))
                              (< (cdr (assoc 70 O)) 4))
                         (and (not O) (findfile (strcat BLOC ".dwg"))))
                     BLOC (alert (if O (REFX) (RUTES)))))
        BLOC (if BLOC (strcase BLOC) (exit))
        N (strcat BLOC M) W T)
  (while W
    (setq O (getpoint "\nPrecise punto de inserción: "))
    (foreach P (if 2D '("X" "Y") '("X" "Y" "Z"))
      (setq A (getpoint O (strcat "\nSitue el extremo de un segmento +"  P
                                  " desde el punto de inserción: "))
            W (getdist O (strcat "\n   Longitud de este segmento en el bloque "
                                 BLOC  " <1>: "))
            W (if W W 1))
      (set (read P) (mapcar '(lambda (CO CA) (+ CO (/ (- CA CO) W))) O A))
      (set (read (strcat "O" P)) (/ (distance O A) W)))
    (setq OZ (if 2D OX OZ)
          W (if (equal (setq UV (U*V (mapcar '- X O) (mapcar '- Y O))) '(0 0 0) Q0)
                " e Y están alineados."
                (if (and (not 2D) (equal (apply '(lambda (X Y Z) (+ X Y Z))
                                                (mapcar '* UV (mapcar '- Z O))) 0
                                         Q0))
                    ", Y y Z son coplanarios.")))
    (if W (alert (strcat "\nREPITE:\nEl punto de Inserción,\nX" W))))
  (setq Z (trans (if 2D (mapcar '+ O UV) Z) 1 0)
        Z-XY Y  Y (trans Y 1 0))
  (QUASI-SENSE-RASTRE) ; ALTERNATIVA: suprimir esta línea, y activar todas las que
;;   (SENSE-RASTRE)    ;              comienzan con ";; " borrando estos caracteres
  (setvar "CMDECHO" 0)
  (command "OSMODE" 0
           "ATTDIA" 0
           "SCP" "3" O X Z-XY)
  (setq O '(0 0 0) X (list OX 0 0) Y (trans Y 0 1) Z (trans Z 0 1)
        Z-XY (list (car Z) (cadr Z) 0) I (if (< (last Z) 0) -1 1)
        NORM-XY (equal (setq X-Y (expt (distance X Y) 2))
                       (setq X-O-Y (+ (expt OX 2) (expt OY 2))) Q0)
```

```
      NORM-Z (equal Z-XY O Q0)
      BL BLOC BLOC (if (tblsearch "BLOCK" BLOC) BLOC (DIBUIX)))
  (SEGR-ATRIBS)
  (if (and NORM-XY NORM-Z)
      (progn
        (eval (append '(command "INSERT" BLOC O "XYZ" OX OY (* OZ I) 0)
                      (if ATREQ (append WWAA-1 WW))))
        (if OO-4 (INSERT** BL OX OY (* OZ I) 0)))
      (INSERT*))
  (command "SCP" "PR"
           "INSNAME" BL
           "ATTDIA" ATDIA
           "OSMODE" OSN
           "DESHACER" "F")
  (setvar "CMDECHO" ECO)
  (princ))

(defun ALINEAR+ (/ O PX PY PZ QX QY QZ D PYD PZD PYDU PZDU PZDXY PZDG PZDGU)
  (setq O '( 0 0 0) PX '(1 0 0) PY '(0 1 0) PZ '(0 0 1)
        QX (trans PX 0 1) QY (trans PY 0 1)
        QZ (trans PZ 0 1) ; no cal usar QZU (trans QZ 1 0), perquè equival a PZ
        D (mapcar '- QX PX) PYD (mapcar '+ PY D) PZD (mapcar '+ PZ D)
        PYDU (trans PYD 1 0) PZDU (trans PZD 1 0))
  (command "DESPLAZA" "P" "" PX QX
           "SCP" "3" QX QY (if (equal QY PYD Q0) PZD PYD))
  (setq PYD (trans PYDU 0 1) PZD (trans PZDU 0 1)
        PZDXY (list (car PZD) (cadr PZD) 0)
        PZDG (polar O (- (angle O PZDXY) (angle O PYD)) (distance O PZDXY))
        PZDG (list (car PZDG) (cadr PZDG) (last PZD))
        PZDGU (trans PZDG 1 0))
  (command "GIRA" "P" "" O "R" O PYD 0
           "SCP" "DE" (list (/ (sqrt 2) 2) 0 0)
           "SCP" "Y" 90
           "GIRA" "P" "" O "R" O (trans PZDGU 0 1) (trans PZ 0 1)
           "SCP" "PR" "SCP" "PR" "SCP" "PR"))

 (defun C:BLOQUEOK (/ Q0 BLOC I SA SS ULT REDEF CAPA ECO OSN AFL SCP)
  (while (not BLOC)
         (setq BLOC (getstring "\nIndique nombre de bloque o [?]: " T))
         (while (= (substr BLOC 1 1) " ")
                (setq BLOC (substr BLOC 2)))
         (if (> (setq I (strlen BLOC)) 0)
             (while (= (substr BLOC I 1) " ")
                    (setq I (1- I) BLOC (substr BLOC 1 I))))
         (setq BLOC (if (wcmatch BLOC (strcat ",*[" (chr 1) "-" (chr 31)
                                              "]*,*\"*,*`**,*`,*," "*/*,*[" (chr 58)
                                              "-" (chr 62) "]*,*?`?*,*`??*,"
                                              "*\\*,*``*,*|*,*[" (chr 127) "-"
                                              (chr 160) "]*"))
                        (prompt "\nNombre bloque no válido.")
                        BLOC)))
  (setq Q0 0.001
        ECO (getvar "CMDECHO"))
  (QUASI-SENSE-RASTRE) ; ALTERNATIVA: suprimir esta línea, y activar todas las que
;;   (SENSE-RASTRE)    ;              comienzan con ";; " borrando estos caracteres
  (if (= BLOC "?")
```

```
    (command "BLOQUE" "?" (progn
                            (prompt "\nIndique bloque(s) a enumerar <*>: ")
                            (setvar "CMDECHO" 1) PAUSE))
    (if (or (not (tblsearch "BLOCK" BLOC))
            (progn
              (initget "Si No")
              (setq REDEF (getkword (strcat "\nEl bloque \"" BLOC "\" ya existe. "
                                            "¿Desea volver a definirlo? [Sí/No] <N>: ")))
              (= REDEF "Si")))
        (progn
          (setq CAPA (getvar "CLAYER")
                OSN (getvar "OSMODE")
                AFL (getvar "AFLAGS")
                I (initget 1) I (getpoint "\nPrecise punto base de inserción: "))
                SA (ssget) SS (ssadd) ULT (entlast))
          (command "CLAYER" "0"
                   "OSMODE" 0
                   "AFLAGS" 9
                   "COPIA" SA "" "0,0,0" ""
                   "ATRDEF" "" "BLOQUEOK" "" "" I "" ""
                   "DESIGNA" (entlast) SA ""
                   "AFLAGS" AFL
                   "SCP" "DE" I)
          (while ULT (if (setq ULT (entnext ULT)) (ssadd ULT SS)))
          (ssdel (entlast) SS)
          (setq SCP (= (getvar "WORLDUCS") 0))
          (if SCP
              (progn
                (ALINEAR+)
                (command "SCP" "")))
          (eval (append '(command "BLOQUE" BLOC)
                        (if REDEF '("S"))
                        '("0,0,0" "P" "")))
          (if SCP (command "SCP" "PR"))
          (command "SCP" "PR"
                   "BORRA" SS ""
                   "AFLAGS" AFL
                   "OSMODE" OSN
                   "CLAYER" CAPA))))
  (command "DESHACER" "F")
  (setvar "CMDECHO" ECO)
  (princ))
```

3. Conclusiones

Dada su naturaleza instrumental, creo que es el lector quien debería sacar sus propias conclusiones del trabajo: cópiese el código precedente en un archivo <***nombre_archivo***>***.lsp***, como texto sin formato, éntrese en AutoCAD, evalúese la expresión **(load "**<***nombre_archivo***>**")** i ejecútese ***INS2D*** o ***INS3D***.

4. Bibliografía

Por heterodoxo que pueda parecer, este trabajo es resultado exclusivo de la aplicación *ad nauseam* del método ensayo/error, con centenares de horas quemadas frente al ordenador. Tal método distará de ser ejemplar, pero es el mío. Fuera de consultas esporádicas a manuales de AutoCAD, no he utilizado directamente fuentes bibliográficas. Subrayo el adverbio de modo porque es evidente que hay un *background* en el sentido del aforismo que afirma de la cultura que "es lo que queda cuando uno ha olvidado lo que aprendió", pero no he llegado a sentir la necesidad de recurrir a fuentes específicas.